



(11) **EP 1 325 409 B1**

(12) **EUROPEAN PATENT SPECIFICATION**

(45) Date of publication and mention
of the grant of the patent:
31.08.2011 Bulletin 2011/35

(21) Application number: **01924591.9**

(22) Date of filing: **28.03.2001**

(51) Int Cl.:
H04L 29/08 (2006.01)

(86) International application number:
PCT/US2001/010644

(87) International publication number:
WO 2002/021262 (14.03.2002 Gazette 2002/11)

(54) **A SHARED FILE SYSTEM HAVING A TOKEN-RING STYLE PROTOCOL FOR MANAGING META-DATA**

GETEILTES DATEISYSTEM MIT EINEM TOKEN-RING-ARTIGEN PROTOKOLL ZUM VERWALTEN
VON METADATEN

SYSTEME DE FICHER PARTAGE A PROTOCOLE DE STYLE ANNEAU EN JETON DESTINE A
GERER DES METADONNEES

(84) Designated Contracting States:
**AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU
MC NL PT SE TR**

(30) Priority: **07.09.2000 US 657079**

(43) Date of publication of application:
09.07.2003 Bulletin 2003/28

(73) Proprietor: **Harmonic Inc.**
San Jose, CA 95134 (US)

(72) Inventor: **MITARU, Alexandru**
Beaverton, OR 97007 (US)

(74) Representative: **Korenberg, Alexander Tal et al**
Kilburn & Strode LLP
20 Red Lion Street
London WC1R 4PJ (GB)

(56) References cited:
US-A- 5 600 834 US-A- 5 758 150

- **STEVENSON D: "Token-based consistency of replicated servers" INTELLECTUAL LEVERAGE. SAN FRANCISCO, FEB. 27 - MAR. 3, 1989, COMPUTER SOCIETY INTERNATIONAL CONFERENCE (COMPCON), WASHINGTON, IEEE COMP. SOC. PRESS, US, vol. CONF. 34, 27 February 1989 (1989-02-27), pages 179-183, XP010014743 ISBN: 0-8186-1909-0**

- **"THE LOCUS DISTRIBUTED FILESYSTEM" LOCUS DISTRIBUTED SYSTEM ARCHITECTURE, XX, XX, 1985, pages 29-72, XP000748057**
- **SHINKAI Y ET AL: "HAMFS file system", RELIABLE DISTRIBUTED SYSTEMS, 1999. PROCEEDINGS OF THE 18TH IEEE SYMPOSIUM ON LAUSANNE, SWITZERLAND 19-22 OCT. 1999, LOS ALAMITOS, CA, USA, IEEE COMPUT. SOC, US, 19 October 1999 (1999-10-19), pages 190-201, XP010356993, ISBN: 978-0-7695-0290-8**
- **GUYENNET H ET AL: "A new consistency protocol implemented in the CALiF system", HIGH-PERFORMANCE COMPUTING, 1997. PROCEEDINGS. FOURTH INTERNATIONAL CONFERENCE ON BANGALORE, INDIA 18-21 DEC. 1997, LOS ALAMITOS, CA, USA, IEEE COMPUT. SOC, US, 18 December 1997 (1997-12-18), pages 82-87, XP010255676, ISBN: 978-0-8186-8067-0**
- **SHINKAI Y ET AL: "HAMFS file system", RELIABLE DISTRIBUTED SYSTEMS, 1999. PROCEEDINGS OF THE 18TH IEEE SYMPOSIUM ON LAUSANNE, SWITZERLAND 19-22 OCT. 1999, LOS ALAMITOS, CA, USA, IEEE COMPUT. SOC, US, 19 October 1999 (1999-10-19), pages 190-201, XP010356993, ISBN: 978-0-7695-0290-8**
- **GUYENNET H ET AL: "A new consistency protocol implemented in the CALiF system", HIGH-PERFORMANCE COMPUTING, 1997. PROCEEDINGS. FOURTH INTERNATIONAL CONFERENCE ON BANGALORE, INDIA 18-21 DEC. 1997, LOS ALAMITOS, CA, USA, IEEE COMPUT. SOC, US, 18 December 1997 (1997-12-18), pages 82-87, XP010255676, ISBN: 978-0-8186-8067-0**

Note: Within nine months of the publication of the mention of the grant of the European patent in the European Patent Bulletin, any person may give notice to the European Patent Office of opposition to that patent, in accordance with the Implementing Regulations. Notice of opposition shall not be deemed to have been filed until the opposition fee has been paid. (Art. 99(1) European Patent Convention).

EP 1 325 409 B1

Description

BACKGROUND INFORMATION

[0001] This invention is generally related to shared file systems and techniques for managing meta-data in such file systems.

[0002] Electronic storage typically features a file system program which implements a set of functions related to the storage and retrieval of data. The file system program associates the physical structures in which files and directories are stored, for instance one or more direct access devices (DADs) such as hard disk drives, with logical structures such as file names. In this way, data can be retrieved using the file name, without having to know the physical location of the data.

[0003] To allow multiple users access to the same user data, a shared file system program allows client software running on each one of a number of computers to write to or read from the same file stored in one or more DADs. The computers and DADs are connected by high-speed links. For instance, each of the computers and the DADs may be agents on a bus, such as a Fibre Channel (FC) bus. Each computer may communicate with the DADs using the Small Computer System Interface (SCSI) protocol. When a computer instructs the DAD to read or write data it is the "initiator" of the request, and the DAD is the "target". The Fibre Channel bus allows multiple "initiators" to coexist on the same bus. This allows the computers to access the same data stored on the DADs, i.e. allows the computers to implement a "shared file system".

[0004] A file system needs two classes of meta-data which provide, (1) a description of the available (free) space on the DADs and (2) a description of the users' data (files). A file system uses meta-data to provide information about and/or documentation of user data managed by the file system. Meta-data may document data about (1) elements or attributes (name, size, data type, etc.), and (2) where the user data is located, how it is associated, its ownership, etc. Meta-data may include descriptive information about the context, quality and condition, or characteristics of the user data.

[0005] The meta-data for a file may include (1) the file name, including the file's path in a hierarchical directory, (2) file attributes, ownership, security descriptors, and (3) descriptors of the DAD addresses where the user data is stored. Whenever a file is written to, renamed, or deleted, its meta-data changes.

[0006] For acceptable performance in accessing files in a shared file system, each computer maintains a copy of the file system meta-data in its own internal solid state random access memory (RAM). Any changes made to the file system meta-data need to be carefully managed across all of the computers, so that each computer has a consistent view of the file system. In essence, (1) a computer needs to lock the file system meta-data (the whole meta-data or just specific portions) in order to gain unique control of it; (2) change it in order to perform the

specific operation (create a file, delete a file, write data into a file, etc), (3) share the changes with all other computers so each computer maintains in its RAM the same image of the meta-data and (4) unlock the file system meta-data to allow other computers to perform their own operation.

[0007] One way for managing access to and thereby maintaining the consistency of meta-data in a shared file system is to provide locking and unlocking primitives using the bus that all computers share to access the DADs. In the Fibre Channel system (running the SCSI protocol) described above, a computer that wishes to change the meta-data attempts to lock a semaphore stored in the memory of one of the DADs. If successful, the computer is allowed to change the meta-data (or a specific portion thereof associated with the particular semaphore). The computer will share the changes with all other computers either by writing the changes to a DAD and instructing the other computers to read them, or by using a network protocol to communicate directly its changes. Finally, the semaphore is unlocked after the update to the meta-data has been reflected in each computer. This technique, however, can hamper the performance of the shared file system, and, in cases of high file system activity, such as during the creation and deletion of a large number of files, considerable delays may be encountered during the locking-unlocking procedure.

[0008] Another technique for maintaining the consistency of meta-data uses a reliable multicast or broadcast protocol, to update the copies of the meta-data in each computer. Once again, however, this technique may not scale well when more than a few computers are participating, as meta-data management tends to diminish the performance of the file system as a whole.

[0009] Further, in David Stevenson, "Token-Based Consistency of Replicated Servers," Intellectual Leverage, San Francisco, Feb 27-Mar. 3, 1989, Computer Society International Conference (COMPCON), Washington, IEEE Com. Soc. Press, US, vol. CONF. 34, 27 February 1989 (1989-02-27) pages 179-183, there is disclosed a system comprising several servers, whereby tokens are used to maintain consistency between the servers.

[0010] In "The LOCUS Distributed Filesystem," pages 29-72, a distributed system architecture is described. Elements of the file system are its distributed naming catalogue, the accompanying synchronisation facilities, integrated replicated storage support, and the mechanisms to allow partitioned operation.

[0011] In Shinkai Y; Tsuchiya Y; Murakami T; Williams J: "HAMFS file system" Reliable Distributed Systems, 1999, Proceedings of the 18th IEEE Sympdsium on Lausanne, Switzerland 19-22 October. 1999, 19 October 1999 (1999-10-19), pages 190-201, there is disclosed a Highly Available Multi-server File System for large commercial UNIX environments.

[0012] Guyennet H et al: "A new consistency protocol implemented in the CALiF system" High-Performance

Computing, 1997. Proceedings. Fourth International Conference on Bangalore, India 18-21 Dec. 1997, Los Alamitos, CA, USA, IEEE Comput. Soc, US, 18 December 1997 (1997-12-18), pages 82-87, discloses a net consistency protocol for distributed shared memory where different shared objects are replicated at each site. This system uses distributed shared memory to transparently handle data sharing using a token technique. Updates of shared data are carried through a virtual ring on the token.

[0013] According to the invention, there is provided a system as defined by claim 1, an article of manufacture as defined by claim 5, and a method as defined by claim 9.

[0014] Further advantageous features and aspects of the invention are described in the subclaims.

[0015] According to an embodiment of the invention, a shared file system is disclosed having one or more shared storage nodes and a number of processing nodes. The processor nodes are connected to each other in a logical ring and to the shared storage nodes. The storage nodes are used to store user data and file system meta-data (FSMD). Each respective processing node has a processor and a memory. The memory contains a number of instructions which, when executed by the processor, cause the respective processing node to (1) disallow modifications to the FSMD until it receives a token, (2) update the FSMD based upon the content of the token received from one of the processing nodes, (3) perform its own changes to the FSMD, (4) append information to the token that describes these changes, and then (5) send the token to the next processing node in the logical ring. Such a mechanism performs two functions in a shared file system: (1) the locking of the meta-data (or a portion of it) and (2) the sharing and updating in each computer an image or copy of the meta-data. Such a scheme has several advantages. For instance, the method may be implemented using relatively high-level software protocols, thus obviating the need for hardware support such as low level networking primitives that implement locking and unlocking of a bus to which the storage nodes are coupled. In addition, there is no need for a reliable multicast or reliable broadcast protocol, thereby providing the potential to scale well when more than a few computers are participating in the shared file system.

[0016] For the particular embodiment in which the changes to the FSMD are journaled in non-volatile memory at each processing node, the file system need not update the FSMD to the storage node very often, thereby further improving the performance of the file system while at the same time providing a reliable system in the event of a catastrophic power failure.

[0017] A token is a packet of data (of variable length) that is sent from one computer to another. The computers that cooperate in implementing a shared file system establish a logical ring, i.e. a logical sequence for sending this token from one computer to another. For example, if there are three computers A, B, and C, a token can be sent on this logical ring from computer A to computer B,

from computer B to computer C and from computer C to computer A. The token performs two functions in the implementation of the shared file system: (1) it allows the locking of file system meta-data, i.e. a computer is allowed to modify the file system meta-data only if it owns the token (the token may be parked in the computer's internal memory) and (2) by appending the descriptions of the modifications to the file system meta-data that each computer performs, the token allows sharing the updates between all computers.

BRIEF DESCRIPTION OF THE DRAWINGS

[0018] The invention is illustrated by way of example and not by way of limitation in the figures of the accompanying drawings in which like references indicate similar elements. It should be noted that references to "an" embodiment in this disclosure are not necessarily to the same embodiment, and they mean at least one.

Fig. 1 illustrates a block diagram of a shared storage system according to an embodiment of the invention.

Fig. 2 depicts a flow diagram of operations for managing meta-data in a shared storage system.

Fig. 3 shows an exemplary format for a token used for managing meta-data in a shared storage system.

DETAILED DESCRIPTION OF THE INVENTION

[0019] Fig. 1 illustrates a block diagram of a system of processing and storage nodes configured according to an embodiment of the invention. Storage nodes 104a and 104b are provided to store user data and the file system meta-data (FSMD). Reference to FSMD or meta-data here means either the entire meta-data of the shared file system, or a portion thereof. Although only two storage nodes are shown, the invention is not limited to any particular number of such nodes. The user data is stored as named files that are part of a hierarchical file system. The term "files" is generically used here to include any data the user wishes to store and retrieve later using its given name. The actual storage hardware in each storage node may be, for instance, a redundant array of inexpensive disks (RAID) comprising one or more DADs.

[0020] Each storage node 104 is coupled to a number of processing nodes 108(1), 112(2),...116(N). Each processing node includes a processor and a memory (not shown) configured according to conventional computing architectures. The memory contains a number of instructions which, when executed by the processor, cause the respective processing node to perform certain operations suitable for managing meta-data. These will be discussed below in connection with the flow diagram of Fig. 2. Each processing node may include, in a further embodiment, non-volatile memory in the forms of solid state semiconductor devices, rotating magnetic disks, or other suitable non-volatile medium, to store information that is needed to re-boot the node or the entire shared file sys-

tem after a catastrophic power failure. Finally, the processing nodes are coupled to the storage nodes 104 through any conventional electronic or optical data interconnect. In a particular embodiment, the interconnect may feature a channel/network standard such as Fibre Channel as described by the Fibre Channel Industry Association. However, other future developments in interconnect technology may also be applicable.

[0021] The methodology for managing meta-data calls for each processing node to store a "copy" of the FSMD. The references to different "copies" here does not necessarily mean that the versions of the FSMDs are at all times identical. The copies of the FSMDs in the processing and storage nodes may be different until changes made by one or more of the processing nodes to the file system meta-data are reflected in all nodes. The initial copy may be delivered to each processing node upon startup, from a copy in the storage nodes 104.

[0022] Each processing node is to update its copy of the FSMD based upon the content of a token received from one of the other processing nodes. The token 124 is generically understood here as a packet or amount of data that may, according to any suitable format, contain a payload that in essence describes the modifications to the FSMD performed by the "upstream" processing nodes, i.e. those other nodes which had possession of the token prior to the current node having possession. Thus, when a processing node receives the token, that node updates and thereby makes current its copy of the FSMD, based upon the content of the token.

[0023] A processing node may change its copy of the FSMD only if that node holds the token 124. This allows "synchronization" of changes made to the FSMD, so that each processing node is allowed to change the FSMD in a deterministic manner. Changes to the FSMD that are made by a processing node are described and appended to the token 124 as a journal piece 126. This is done prior to releasing and sending the token 124 to another processing node.

[0024] Note that the accessing of the user data in the shared storage node 104 need not modify the FSMD. Certain file system operations, such as reading user data or overwriting a small amount of user data, do not require any changes to the meta-data, and hence may be performed by a processing node in the absence of the token 124. Also, changes to the FSMD need not involve an access to the storage nodes 104, since the FSMD is kept in the memory of the processing node. The creation of a file changes the FSMD by introducing a new name of the file, file attributes, a date of creation of the file, as well as the size and location of the file in the storage nodes 104. Deleting a file will change the FSMD by freeing a file name from a name space of the shared file system. Writing to a file changes the FSMD by changing the number and/or location of the data blocks in the storage nodes 104 that actually house the user data.

[0025] The FSMD may also include allocation tables. File write operations modify a storage unit allocation ta-

ble, by indicating that certain blocks that were previously free are now used. Thus, the token may also contain a description of changes that have been made to such a block allocation table, where each processing node has a copy of such a block allocation table. For the sake of brevity, any reference to "FSMD" here is also understood perhaps to include a storage unit allocation table.

[0026] Substantially the same operations, including the updating of the copy of the FSMD and adding information to the token that describes further changes to the FSMD, may be performed by the other processing nodes once they are in possession of the token. For instance, the processing node 112(2) may add its journal piece 128, and so on until processing node 116(N).

[0027] Turning now to Fig. 2, and also referring to elements of Fig. 1, a flow diagram of an embodiment of the meta-data management scheme is shown. The flow begins with operation 204 in which a logical/virtual ring is established between all processing nodes having write-access to the shared storage nodes 104. In addition, in operation 208, the token 124 is created where the token is to be sent from one processing node to another in the ring. Operation 212 provides that a processing node may modify the FSMD (i.e. its copy of the FSMD) only if the node holds the token 124. Operation 216 is provided so that changes by a processing node to its copy of the FSMD are journaled at that node.

[0028] The flow continues with operation 220 in which the processing node having the token appends its piece of journal (that was generated while the node had the token) to the token, prior to sending the token to the next node. Thus, in Fig. 1, it can be seen that processing node 108(1) adds an amount of information such as its journal piece 126, referenced "1", to the bottom of a stack of other changes made to the FSMD. The token 124 carries as a payload a journal piece from each processing node at which the token has been and at which a change was made to the FSMD. In Fig. 1, these changes are indicated in the token 124 by reference numerals 2...N, where the first change at the top of the stack was made by the processing node 112(2) the previous time that node had the token 124.

[0029] After the processing node 108(1) has added information describing the new change to the FSMD, the token 124 is sent to the next processing node in the ring, in this case, processing node 112(2). In Fig. 2, operation continues with step 228. When the processing node 112(2) receives the token 124, this node applies to its copy of the FSMD the operations listed in the payload of the token 124, including any and all changes to the FSMD that were made by all other nodes, namely nodes referenced 1 and 3...N. Once updated in this manner, the copy of the FSMD may be used by that node to reliably access the storage nodes 104. In addition to the updating of its copy of the FSMD, the processing node 112(2) deletes the journal piece (referenced "2" at the top of the stack in the payload) that was created by this node a previous time it had the token 124, as in operation 232. This helps

keep the payload size from growing too large. Any new changes to the FSMD by this node are journaled, and the node appends this journal piece 128 to the bottom of the stack, prior to sending the token 124 to the next node. When the processing node 112 (2) forwards the token 124 to the next node in the ring, the operations of Fig. 2 may repeat with operation 216 at the next node. Thus, it can be seen that the token 124 is passed around the ring to give each of the processing nodes a chance, and preferably a fair chance, to change its copy of the FSMD.

[0030] It should be noted that even if a processing node does not have the token 124, it may still be able to access the user data in the storage nodes 104 if doing so does not require any modification to the FSMD. For instance, file read operations generally do not modify the meta-data, and, as such, may be performed by any processing node without requiring possession of the token 124.

[0031] In addition to information concerning changes made to the copy of the FSMD in each processing node, the token payload may also include descriptions of changes made to a block allocation table of the storage nodes 104. To simplify the description of this aspect of the invention, a "change to the FSMD" is also understood as perhaps including a change to the file system block allocation table to indicate which blocks have been freed and which blocks are being used.

[0032] Fig. 3 illustrates an exemplary format for each journal piece 126, 128 in the token payload. Each piece identifies the processing node with which it is associated, using for instance a processing node identification code. In addition, the piece contains a list of all changes made to the FSMD by that node. For instance, such changes may include the file name, its type, whether the file was created, modified, or deleted, the blocks currently used by the file, as well as the currently available free blocks. The latter two items may actually be pointers to locations in the storage nodes 104 that contains or may contain the user data. One of ordinary skill in the art will recognize that a wide range of different formats may be developed to hold the information that describes the changes made to the FSMD by each processing node.

[0033] To summarize, several embodiments of a method and system for managing meta-data in a shared storage system have been disclosed. The method may provide a more scalable and lower cost alternative to certain conventional techniques for managing meta-data including the use of locking-unlocking low level network primitives and/or reliable multicast or broadcast protocols. In the foregoing specification, the invention has been described with reference to specific exemplary embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention as set forth in the appended claims. For instance, the transmission medium and protocol for sending and receiving the token between processing nodes may be the same as the ones used for accessing the user data, or they may be separate for improved performance. The

specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

5 Claims

1. A system comprising: one or more shared storage nodes (104a, 104b) to store user data and file system meta-data "FSMD" and a plurality of processing nodes (108, 112, 116) coupled to the shared storage nodes (104a, 104b), each respective processing node (108, 112, 116) having a processor and a memory, **characterised in that** the memory contains a plurality of instructions which, when executed by the processor, cause the respective processing node to disallow changes to a stored copy of the FSMD unless a token (124) is present, update the copy of the FSMD based upon content of the token (124) received from one of the plurality of processing nodes (108, 112, 116), add information to the token (124) that describes a change that node makes to the copy of the FSMD, and then send the token to another one of the processing nodes (108, 112, 116), wherein in each respective processing node (108, 112, 116) the memory includes further instructions which, when executed by the processor, cause that node to be part of a virtual ring made of the plurality of processing nodes (108, 112, 116), the token to be passed around the ring to give each of the plurality of processing nodes a fair chance to change a copy of the FSMD.
2. The system of claim 1 wherein in each respective processing node (108, 112, 116), the memory includes further instructions which, when executed by the processor, delete information in the token that describes a change that node made to the copy of the FSMD a previous time that node had the token.
3. The system of claim 1 wherein each respective processing node (108, 112, 116) further comprises non-volatile memory to store a journal that describes changes made to the FSMD.
4. The system of claim 3 wherein in each respective processing node (108, 112, 116), the memory includes further instructions which, when executed by the processor, cause that node to write the journal to the shared storage nodes when the non-volatile memory is close to being full.
5. An article of manufacture suitable for use as part of a shared storage system having one or more shared storage nodes (104a, 104b) to store user data and file system meta-data "FSMD" and a plurality of processing nodes (108, 112, 116) coupled to the shared storage node (104a, 104b), each respective processing node (108, 112, 116) having a processor,

characterised in that the article of manufacture comprises: a machine-readable medium containing a plurality of instructions which, when executed by the processor, cause the respective processing node (108, 112, 116) to disallow changes to a stored copy of the FSMD unless a token (124) is present, update the copy of the FSMD based upon content of the token (124) received from one of the plurality of processing nodes (108, 112, 116), add information to the token (124) that describes a change that node makes to the copy of the FSMD, and then send the token to another one of the processing nodes (108, 112, 116), wherein the machine readable medium includes further instructions which, when executed by the processor, cause that node to be part of a virtual ring made of the plurality of processing nodes (108, 112, 116), the token to be passed around the ring to give each of the plurality of processing nodes a fair chance to change a copy of the FSMD.

6. The article of manufacture of claim 5 wherein the machine readable medium includes further instructions which, when executed by the processor, delete information in the token (124) that describes a change that node made to the copy of the FSMD a previous time that node had the token.
7. The article of manufacture of claim 5 wherein the machine readable medium includes a non-volatile portion to store a journal that describes changes made to the copy of the FSMD.
8. The article of manufacture of claim 7 wherein the machine readable medium includes further instructions which, when executed by the processor, cause that node to write the journal to the shared storage nodes when the non-volatile memory is close to being full.
9. A method for managing file system meta-data "FSMD" in a processing node of a system comprising one or more shared storage nodes (104a, 104b) to store user data and FSMD; and a plurality of processing nodes (108, 112, 116) coupled to the shared storage nodes, said method **characterised by** comprising the steps of: disallowing changes to a stored copy of the FSMD unless a token (124) is present; updating the stored copy of the FSMD based upon content of the token received from one of a plurality of processing nodes (108, 112, 116); adding information to the token (124) that describes a change the processing node (108, 112, 116) makes to the copy of the FSMD; and then sending the token to another one of the processing nodes (108, 112, 116); the method further comprising: forming a virtual ring made of the plurality of processing nodes (108, 112, 116), the token to be passed around the ring to give each of the plurality of processing nodes a fair

chance to change the FSMD.

10. The method of claim 9 further comprising: deleting information in the token (124) that describes a change the processing node made to the copy of the FSMD a previous time that node had the token.
11. The method of claim 9 further comprising storing a journal that describes changes made to the FSMD in non-volatile memory.
12. The method of claim 11 further comprising: writing the journal to the storage nodes when the non-volatile memory is close to being full.

Patentansprüche

1. System, aufweisend: einen oder mehrere gemeinsam genutzte Speicherknoten (104a, 104b) zum Speichern von Benutzerdaten und Dateisystem-Metadaten "FSMD" und mehrere mit den gemeinsam genutzten Speicherknoten (104a, 104b) verbundene Verarbeitungsknoten (108, 112, 116), wobei jeder entsprechende Verarbeitungsknoten (108, 112, 116) einen Prozessor und einen Speicher besitzt, **dadurch gekennzeichnet, dass** der Speicher mehrere Instruktionen enthält, welche, wenn sie durch den Prozessor ausgeführt werden, den entsprechenden Verarbeitungsknoten veranlassen: Änderungen an einer gespeicherten Kopie der FSMD zu verbieten, sofern nicht ein Token (124) vorhanden ist, die Kopie der FSMD auf der Basis von Inhalt des von einem der mehreren Verarbeitungsknoten (108, 112, 116) empfangenen Tokens (124) zu aktualisieren, Information dem Token (124) hinzuzufügen, die eine Änderung beschreibt, die der Knoten an der Kopie der FSMD vornimmt, und dann den Token an einen anderen von den Verarbeitungsknoten (108, 112, 116) zu senden, wobei in jedem entsprechenden Verarbeitungsknoten (108, 112, 116) der Speicher ferner Instruktionen enthält, welche, wenn sie durch den Prozessor ausgeführt werden, bewirken, dass der Knoten Teil eines von den mehreren Verarbeitungsknoten (108, 112, 116) gebildeten virtuellen Rings ist, wobei der Token um den Ring herum weiterzugeben ist, um jedem von den mehreren Verarbeitungsknoten eine faire Chance zu geben, eine Kopie der FSMD zu verändern.
2. System nach Anspruch 1, wobei in jedem entsprechenden Verarbeitungsknoten (108, 112, 116) der Speicher ferner Instruktionen enthält, welche, wenn sie durch den Prozessor ausgeführt werden, Information in dem Token löschen, die eine Änderung beschreibt, die der Knoten an der Kopie der FSMD zu einem früheren Zeitpunkt vorgenommen hatte, an dem der Knoten den Token hatte.

3. System nach Anspruch 1, wobei jeder entsprechende Verarbeitungsknoten (108, 112, 116) ferner einen nicht-flüchtigen Speicher zum Speichern eines Journals aufweist, das die an den FSMD vorgenommenen Änderungen beschreibt. 5
4. System nach Anspruch 3, wobei in jedem entsprechenden Verarbeitungsknoten (108, 112, 116) der Speicher ferner Instruktionen enthält, welche, wenn sie durch den Prozessor ausgeführt werden, bewirken, dass der Knoten das Journal in die gemeinsam genutzten Speicherknoten schreibt, wenn der nicht-flüchtige Speicher nahezu voll ist. 10
5. Herstellungsgegenstand der zur Verwendung als Teil eines gemeinsam genutzten Speichersystems geeignet ist, das einen oder mehrere gemeinsam genutzte Speicherknoten (104a, 104b) zum Speichern von Benutzerdaten und Dateisystem-Metadaten "FSMD" und mehrere mit den gemeinsam genutzten Speicherknoten (104a, 104b) verbundene Verarbeitungsknoten (108, 112, 116) besitzt, wobei jeder entsprechende Verarbeitungsknoten (108, 112, 116) einen Prozessor besitzt, **dadurch gekennzeichnet, dass** der Herstellungsgegenstand aufweist: ein maschinenlesbares Medium mit mehreren Instruktionen, welche, wenn sie durch den Prozessor ausgeführt werden, den entsprechenden Verarbeitungsknoten (108, 112, 116) veranlassen: Änderungen an einer gespeicherten Kopie der FSMD zu verbieten, sofern nicht ein Token (124) vorhanden ist, die Kopie der FSMD auf der Basis von Inhalt des von einem der mehreren Verarbeitungsknoten (108, 112, 116) empfangenen Tokens (124) zu aktualisieren, Information dem Token (124) hinzuzufügen, die eine Änderung beschreibt, die der Knoten an der Kopie der FSMD vornimmt, und dann den Token an einen anderen von den Verarbeitungsknoten (108, 112, 116) zu senden, wobei das maschinenlesbare Medium ferner Instruktionen enthält, welche, wenn sie durch den Prozessor ausgeführt werden, bewirken, dass der Knoten Teil eines von den mehreren Verarbeitungsknoten (108, 112, 116) gebildeten virtuellen Rings ist, wobei der Token um den Ring herum weiterzugeben ist, um jedem von den mehreren Verarbeitungsknoten eine faire Chance zu geben, eine Kopie der FSMD zu verändern. 20
25
30
35
40
45
6. Herstellungsgegenstand nach Anspruch 5, wobei das maschinenlesbare Medium ferner Instruktionen enthält, welche, wenn sie durch den Prozessor ausgeführt werden, Information in dem Token (124) löschen, die eine Änderung beschreibt, die der Knoten an der Kopie der FSMD zu einem früheren Zeitpunkt vorgenommen hatte, an dem der Knoten den Token hatte. 50
55
7. Herstellungsgegenstand nach Anspruch 5, wobei das maschinenlesbare Medium einen nicht-flüchtigen Abschnitt zum Speichern eines Journals aufweist, das die an der Kopie der FSMD vorgenommenen Änderungen beschreibt.
8. Herstellungsgegenstand nach Anspruch 7, wobei das maschinenlesbare Medium ferner Instruktionen enthält, welche, wenn sie durch den Prozessor ausgeführt werden, bewirken, dass der Knoten das Journal in die gemeinsam genutzten Speicherknoten schreibt, wenn der nicht-flüchtige Speicher nahezu voll ist.
9. Verfahren zum Verwalten von Dateisystem-Metadaten "FSMD" in einem Verarbeitungsknoten eines Systems, das einen oder mehrere gemeinsam genutzte Speicherknoten (104a, 104b), um Benutzerdaten und FSMD zu speichern, und mehrere mit den gemeinsam genutzten Speicherknoten verbundene Verarbeitungsknoten (108, 112, 116) aufweist, wobei das Verfahren **dadurch gekennzeichnet ist, dass** es die Schritte aufweist: Verboten von Änderungen an einer gespeicherten Kopie der FSMD, sofern nicht ein Token (124) vorhanden ist; Aktualisieren der gespeicherten Kopie der FSMD auf der Basis von Inhalt des von einem der mehreren Verarbeitungsknoten (108, 112, 116) empfangenen Tokens; Hinzufügen von Information zu dem Token (124), die eine Änderung beschreibt, die der Verarbeitungsknoten (108, 112, 116) an der Kopie der FSMD vornimmt; und dann Senden des Tokens an einen anderen von den Verarbeitungsknoten (108, 112, 116); wobei das Verfahren ferner den Schritt aufweist: Bilden eines aus den mehreren Verarbeitungsknoten (108, 112, 116) bestehenden virtuellen Rings, wobei der Token um den Ring herum weiterzugeben ist, um jedem von den mehreren Verarbeitungsknoten eine faire Chance zu geben, die FSMD zu verändern.
10. Verfahren nach Anspruch 9, ferner mit dem Schritt: Löschen von Information in dem Token (124), die eine Änderung beschreibt, die der Verarbeitungsknoten an der Kopie der FSMD zu einem früheren Zeitpunkt vorgenommen hatte, an dem der Knoten den Token hatte.
11. Verfahren nach Anspruch 9, ferner mit dem Schritt der Speicherung eines Journals, das an den FSMD vorgenommene Änderungen beschreibt, in einem nicht-flüchtigen Speicher.
12. Verfahren nach Anspruch 11, ferner mit dem Schritt:
Schreiben des Journals in die Speicherknoten, wenn der nicht-flüchtige Speicher nahezu voll ist.

Revendications

1. Système comprenant : un ou plusieurs noeuds de stockage partagé (104a, 104b) pour stocker des données d'utilisateur et des métadonnées de système de fichiers « FSMD » et plusieurs noeuds de traitement (108, 112, 116) couplés aux noeuds de stockage partagé (104a, 104b), chaque noeud de traitement respectif (108, 112, 116) ayant un processeur et une mémoire, **caractérisé en ce que** la mémoire contient une pluralité d'instructions qui, lorsqu'elles sont exécutées par le processeur, amènent le noeud de traitement respectif à interdire les changements d'une copie stockée des FSMD à moins qu'un jeton (124) ne soit présent, mettre à jour la copie des FSMD sur la base du contenu du jeton (124) reçu de l'un des plusieurs noeuds de traitement (108, 112, 116), ajouter des informations au jeton (124) qui décrivent un changement que ce noeud effectue sur la copie des FSMD, puis envoyer le jeton à un autre des noeuds de traitement (108, 112, 116), dans lequel dans chaque noeud de traitement respectif (108, 112, 116) la mémoire comprend d'autres instructions qui, lorsqu'elles sont exécutées par le processeur, amènent ce noeud à faire partie d'un anneau virtuel constitué de la pluralité de noeuds de traitement (108, 112, 116), le jeton devant passer autour de l'anneau pour donner à chacun de la pluralité de noeuds de traitement une chance équitable de changer une copie des FSMD.
2. Système selon la revendication 1, dans lequel dans chaque noeud de traitement respectif (108, 112, 116), la mémoire comprend d'autres instructions qui, lorsqu'elles sont exécutées par le processeur, suppriment des informations dans le jeton qui décrivent un changement que ce noeud a effectué sur la copie des FSMD une fois précédente que le noeud a eu le jeton.
3. Système selon la revendication 1, dans lequel chaque noeud de traitement respectif (108, 112, 116) comprend en outre une mémoire rémanente pour stocker un journal qui décrit les changements apportés aux FSMD.
4. Système selon la revendication 3, dans lequel dans chaque noeud de traitement respectif (108, 112, 116), la mémoire comprend d'autres instructions qui, lorsqu'elles sont exécutées par le processeur, amènent ce noeud à écrire le journal sur les noeuds de stockage partagé lorsque la mémoire rémanente est presque pleine.
5. Article de fabrication apte à être utilisé dans le cadre d'un système de stockage partagé comportant un ou plusieurs noeuds de stockage partagé (104a, 104b) pour stocker des données d'utilisateur et des métadonnées de système de fichiers « FSMD » et plusieurs noeuds de traitement (108, 112, 116) couplés au noeud de stockage partagé (104a, 104b), chaque noeud de traitement respectif (108, 112, 116) ayant un processeur, **caractérisé en ce que** l'article de fabrication comprend : un support lisible par machine contenant une pluralité d'instructions qui, lorsqu'elles sont exécutées par le processeur, amènent le noeud de traitement respectif (108, 112, 116) à interdire les changements d'une copie stockée des FSMD à moins qu'un jeton (124) ne soit présent, mettre à jour la copie des FSMD sur la base du contenu du jeton (124) reçu de l'un des plusieurs noeuds de traitement (108, 112, 116), ajouter des informations au jeton (124) qui décrivent un changement que ce noeud effectue sur la copie des FSMD, puis envoyer le jeton à un autre des noeuds de traitement (108, 112, 116), dans lequel le support lisible par machine comprend d'autres instructions qui, lorsqu'elles sont exécutées par le processeur, amènent ce noeud à faire partie d'un anneau virtuel constitué de la pluralité de noeuds de traitement (108, 112, 116), le jeton devant passer autour de l'anneau pour donner à chacun de la pluralité de noeuds de traitement une chance équitable de changer une copie des FSMD.
6. Article de fabrication selon la revendication 5, dans lequel le support lisible par machine comprend d'autres instructions qui, lorsqu'elles sont exécutées par le processeur, suppriment des informations dans le jeton (124) qui décrivent un changement que ce noeud a effectué sur la copie des FSMD une fois précédente que le noeud a eu le jeton.
7. Article de fabrication selon la revendication 5, dans lequel le support lisible par machine comprend en outre une partie rémanente pour stocker un journal qui décrit les changements apportés à la copie des FSMD.
8. Article de fabrication selon la revendication 7, dans lequel le support lisible par machine comprend d'autres instructions qui, lorsqu'elles sont exécutées par le processeur, amènent ce noeud à écrire le journal sur les noeuds de stockage partagé lorsque la mémoire rémanente est presque pleine.
9. Procédé de gestion de métadonnées de système de fichiers « FSMD » dans un noeud de traitement d'un système comprenant un ou plusieurs noeuds de stockage partagé (104a, 104b) pour stocker des données d'utilisateur et des FSMD ; et plusieurs noeuds de traitement (108, 112, 116) couplés aux noeuds de stockage partagé, ledit procédé étant **caractérisé en ce qu'il** comprend les étapes consistant à : interdire les changements d'une copie stockée des FSMD à moins qu'un jeton (124) ne soit

présent ; mettre à jour la copie stockée des FSMD sur la base du contenu du jeton reçu de l'un des plusieurs noeuds de traitement (108, 112, 116) ; ajouter des informations au jeton (124) qui décrivent un changement que ce noeud de traitement (108, 112, 116) effectue sur la copie des FSMD ; puis envoyer le jeton à un autre des noeuds de traitement (108, 112, 116) ; le procédé comprenant en outre l'étape consistant à former un anneau virtuel constitué de la pluralité de noeuds de traitement (108, 112, 116), le jeton devant passer autour de l'anneau pour donner à chacun de la pluralité de noeuds de traitement une chance équitable de changer les FSMD.

10. Procédé selon la revendication 9, comprenant en outre l'étape consistant à supprimer des informations dans le jeton (124) qui décrivent un changement que le noeud de traitement a effectué sur la copie des FSMD une fois précédente que le noeud a eu le jeton.
11. Procédé selon la revendication 9, comprenant en outre l'étape consistant à stocker un journal qui décrit les changements apportés aux FSMD dans une mémoire rémanente.
12. Procédé selon la revendication 11, comprenant en outre l'étape consistant à écrire le journal sur les noeuds de stockage lorsque la mémoire rémanente est presque pleine.

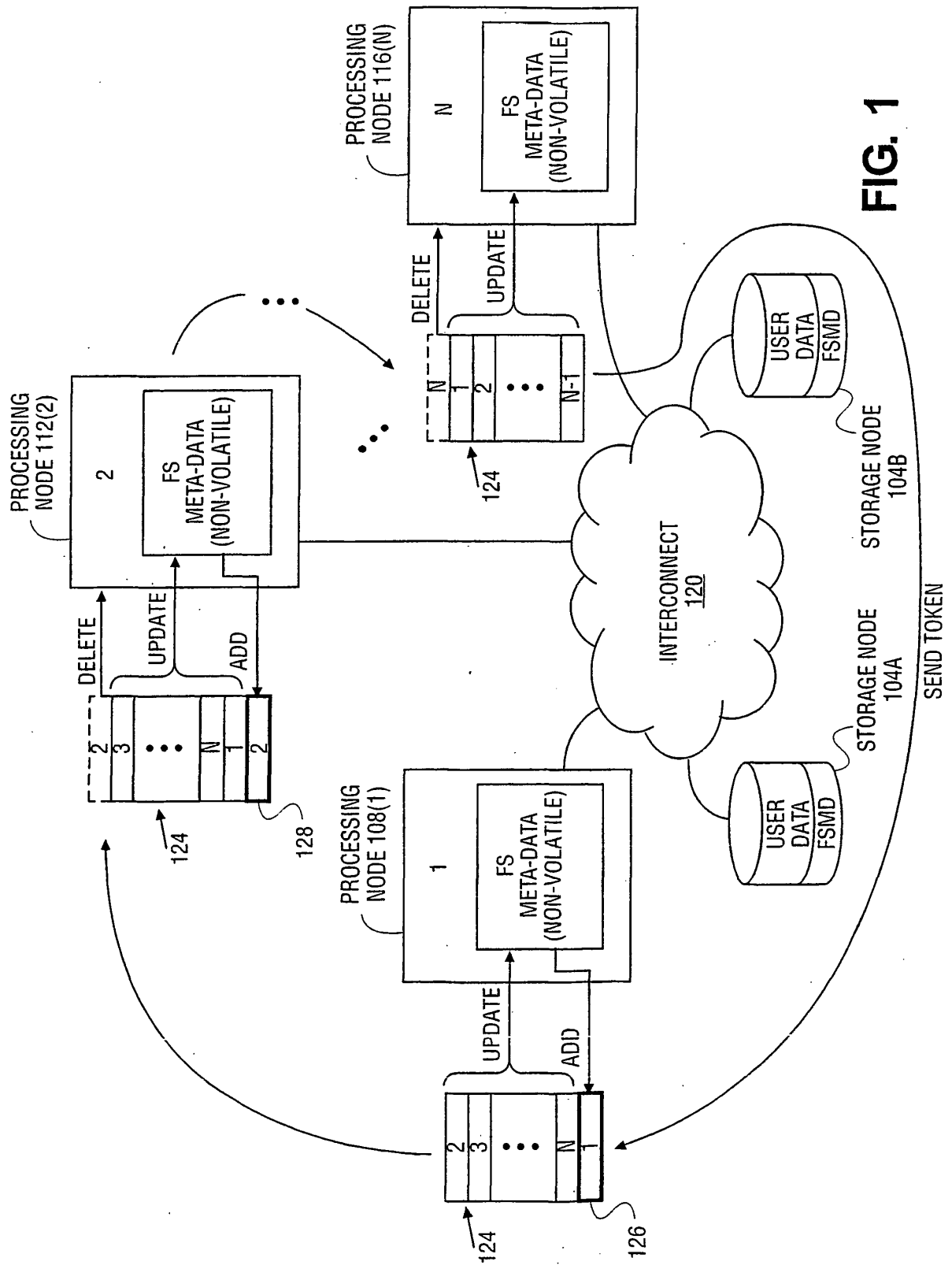


FIG. 1

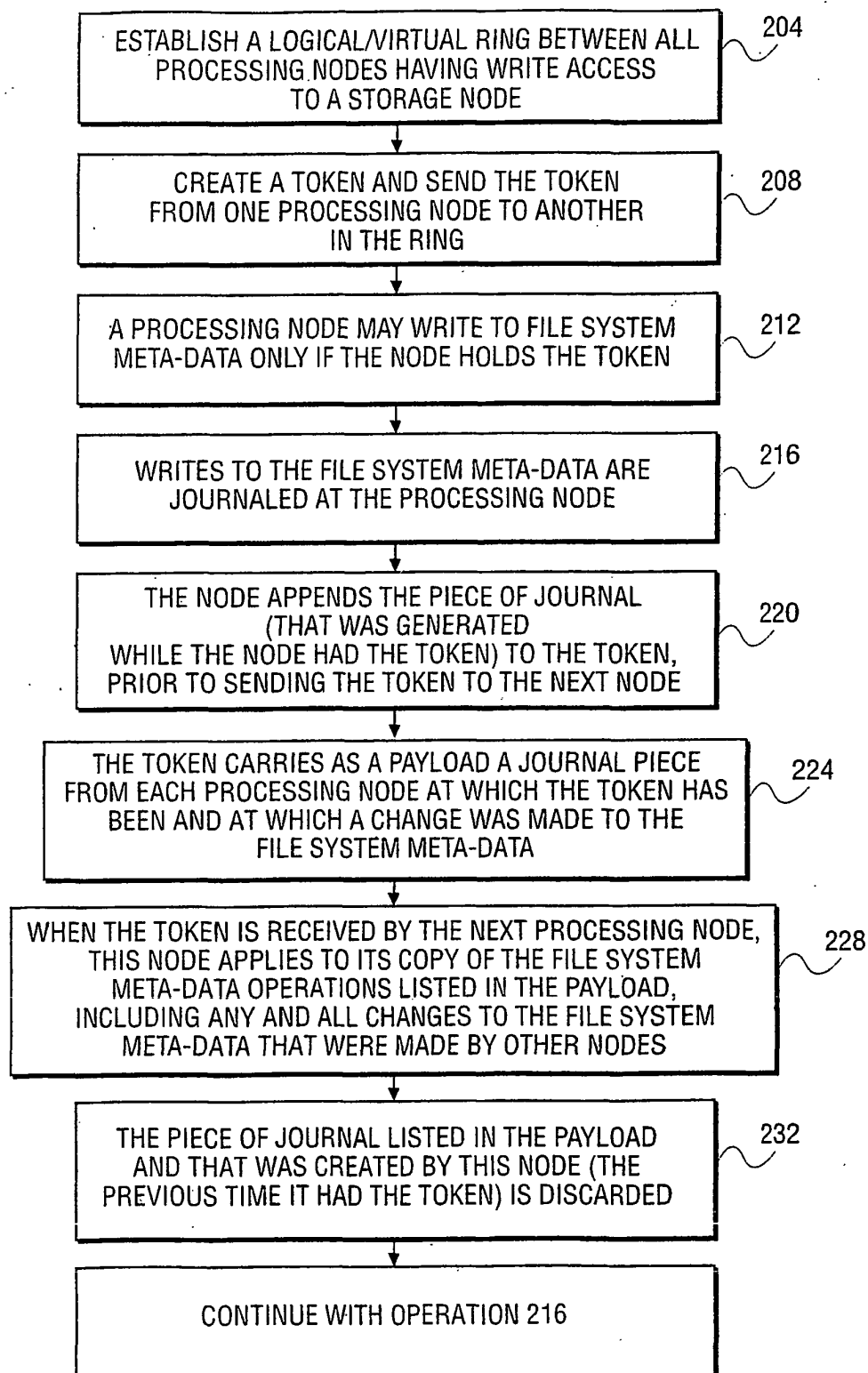


FIG. 2

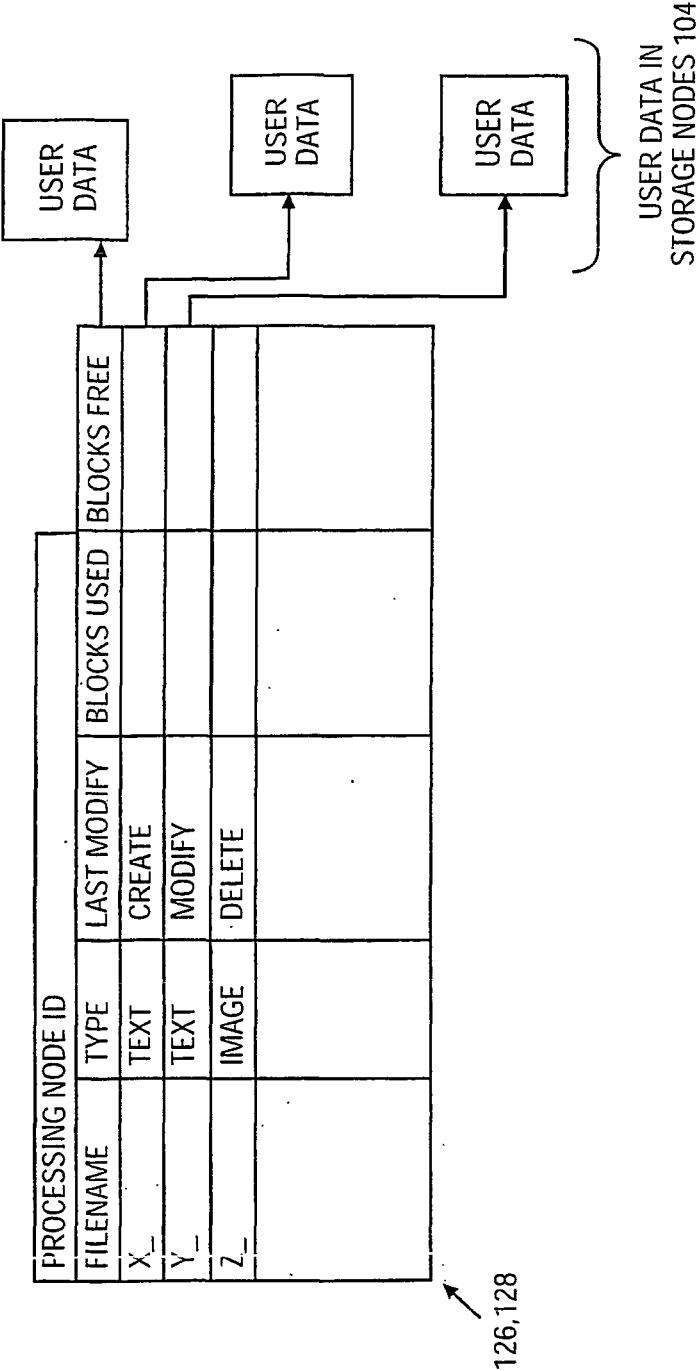


FIG. 3

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Non-patent literature cited in the description

- **David Stevenson.** Token-Based Consistency of Replicated Servers. *Intellectual Leverage*, 27 February 1989 [0009]
- Computer Society International Conference (COMP-CON). IEEE Com. Soc. Press, 27 February 1989, vol. 34, 179-183 [0009]
- *The LOCUS Distributed Filesystem*, 29-72 [0010]
- **Shinkai Y ; Tsuchiya Y ; Murakami T ; Williams J.** HAMFS file system. *Reliable Distributed Systems, 1999, Proceedings of the 18th IEEE Symposium on Lausanne*, 19 October 1999, 190-201 [0011]
- A new consistency protocol implemented in the CALiF system. **Guyennet H et al.** High-Performance Computing, 1997. Proceedings. Fourth International Conference on Bangalore. IEEE Comput. Soc, 18 December 1997, 82-87 [0012]